# Introduction

## Overview

Hierarchical Internet of Things (HIOT) is a communication protocol for the Internet of Things (IOT). It is designed to provide a full suite of messaging for the configuration, management and operational functionality of any IOT use case.

In addition to traditional IOT messaging, HIOT includes message types which are specifically designed to support delegation of control and data aggregation and processing at or near the source of the data. This delegation and local data processing is where HIOT gets its name, enabling IOT to move from a traditional flat structure to a hierarchy of devices and associated roles in order to deliver maximum scalability.

There is also a specific message type aimed at increased interoperability. Handler messages, which allow new functional to be added to any remote device at any time, can be used to integrate with third part components and translate between HIOT and other message types.

During the development of the protocol, a particular use case was selected in order to test the suitability of the specific message types, a “smart” office building. To examine the practical functionality, a partial implementation of the protocol was developed in NodeJS with MQTT as the messaging protocol.

While this document makes numerous references to MQTT throughout, there should no reason why other messaging protocols could not be used in its place.

## Terminology

HIOT in intended to provide full scope communications for any IOT use-cases. In order to illustrate functionality, a number of terms are used to describe specific components. These terms are defined below.

### Platform

Platform refers to typical server-side components offered as part of a SaaS or cloud solution for IOT.

### Deployment

In our selected use case, a smart office building, the term deployment refers to all IOT elements contained within the building.

### Message

Work to date on HIOT has been based on MQTT and a message is any communication sent on an MQTT topic. However, this should be considered non-normative and there should be no reason why other protocols could be used in place of, or in addition to, MQTT.

### Network

In the context of HIOT, the network refers to the underlying network for the messaging protocol. Like MQTT, there is no apparent limitation in terms of what network protocols are used.

### Topic

The term topic is taken from the MQTT specification and is used in that context throughout this document. Channels and Paths for other messaging protocols should be equally applicable and HIOT should be fully implementable using any alternative to MQTT.

### Handler

Handlers are a combination of configuration settings and executables or scripts which can provide additional functionality within a deployment. In our partial NodeJS implementation, handler files are JavaScript files which are Base64 encoded as part of Handler messages sent to specific devices. These handler files could also be executable files, byte-code or any other type of files capable of being executed on a device.

### Cloud-side

“Cloud-side” refers to typical SaaS offerings, including remote administration tools and user-facing components. The cloud-side has been divided into a number of subcomponents offering specific functionality. While some of these components and their functionality is outside of the scope of HIOT, they are included or referenced here for the sake of completeness.

### Path

Paths for part of each topic sent and received by a deployment. For example, a health message would take the form “h/<device path>”, where <device path> is the unique path for the associated device.

Each role deployed to a device also has a unique path. For example, a physical device may have multiple sensors and/or controllers attached, with each one having its own unique path. This facilitates the movement of roles across devices without the need for widespread configuration change.

#### Platform, UI and Coordinator Link

Strictly speaking, the Coordinator Link is the only cloud-side element within the scope of HIOT. The Platform component refers to client or user facing tools, typically exposed through REST web-services and web-sockets. UI refers to any user interface connected to the Platform component.

The Coordinator Link (in conjunction with the deployment-side Coordinator, described below) is responsible for all communications between the Platform and a deployment.

Throughout the document, unless called out at the time, the term Platform refers to all cloud-side elements within IOT.

### Deployment-side

“Deployment-side” refers to the local IOT implementation - for example, our smart building or a production line in a factory. Within the Deployment-side, generic devices are configured via messages sent from the platform and their functionality is determined by their configured roles and associated handler files

#### Device

This is the base implementation for any component within a deployment. A device is capable of receiving configuration data from the platform and can transfer Health messages (described later) and Error messages to the platform.

#### Role: Actuator/controller

Controllers are responsible for interfacing with real world elements of the deployment such as heating and lighting controls etc. Handler files are deployed to devices to implement the required functionality, while the configuration message sent from the deployment defines the required parameters, including a list of all commands and their parameters.

A device can have any number of configured controllers, with each controller being assigned its own messaging topic.

#### Role: Aggregator

Aggregators are a key component in HIOT and allow the deployment to process data locally rather than relying on cloud-side resources. In our selected use case, a number of temperature sensors, distributed across an open-plan area, can transmit readings to an aggregator which can calculate the mean, max and min temperature for the overall area. This data can then be fed to another aggregator which calculates the temperature across the entire floor. Based on the configuration and the handler file, raw sensor data can be included or omitted from the data transferred cloud-side.

#### Role: Broker

Brokers accept commands and forward them to other brokers or controllers. The rationale behind Brokers is to abstract the physical implementation of a deployment and reduce the granularity by grouping related controllers together. In addition, local (deployment-side) Commanders can interact directly with brokers without any cloud-side dependencies.

#### Role: Coordinator

The Coordinator is the only deployment-side device with external connectivity. Each deployment has only one active Coordinator at any time. All messages between the cloud-side and deployment-side pass through the Coordinator, allowing for targeted hardening and security measures and decreasing the deployment’s attack surface.

#### Role: Sensor

Sensors provide real-world data to the deployment and can be with a specified device through the use of configuration data from the Platform and an associated handler files.

A device can have any number of configured sensors, with each sensor having its own messaging topic.

#### Role: Commander

A commander is a local user-interface for the deployment. Unlike the UI component specified as part of the cloud-side, a Commander can communicate directly with specific aggregators and brokers via their messaging topics and subscriptions.

In our selected use-case, a Commander would likely take the form of a wall mounted touchscreen to control and monitor the environment in the immediate vicinity, e.g. control and monitor the temperature in lighting in an open plan office or manage conferencing equipment in a meeting room.

#### Example Implementation

The diagram below illustrates a minimal HIOT implementation.

D: A HIOT Device

S: Sensor – connected directly to a device or integrated via MQTT or another protocol. Configuration data from the Platform and a handler file implement are used to integrate the sensor into HIOT

C: Controllers. Similar to Sensors, these can be directly connected or external and the use of handler files is intended to improve integration and interoperability.

A: Aggregators. Configuration data manage the topic subscriptions (from sensors or other aggregators) and publications (towards the cloud-side). Handler files are deployed to devices to perform operations on the data (e.g. read data from multiple sensors via subscriptions and send the mean, min and max of the reading to the platform via publications). For example, the Sensors connected to the aggregators in the diagram could all be temperature sensors - the first aggregator could compute mean, min and max temperatures for an open plan office, based on the readings from relevant sensors and the next aggregator could calculate the average temperature for a floor/building etc.

B: Brokers act as intermediaries between the platform and the controllers - so the broker in the diagram could control all the lighting/heating controls for the open plan office.

CR: Coordinator - essentially this is the gateway device for a deployment – a single point in and out of the deployment and the only local device which can communicate with the platform. In the event of a failure, this role can be moved to another device through the use of Coordinator messages across the deployment.

CrL: Coordinator link - this sits between the core platform and the deployment and coordinates messages between deployment-side and cloud-side.

DM: This is just a shared data layer used by both the cloud-side Platform and the Coordinator Link. Not relevant to HIOT, but included for completeness.

UI: This is any user facing solution connected to the Platform component.

All of the roles (and other device attributes) are controlled by device-specific configuration files which are managed cloud-side.

#### C:\Users\mhealy\Desktop\g490.png

# Message Structure

MQTT channels use single character, base 58 elements

?/a/b/c/d -- ? = message type

2 levels (a) = 58 devices

3 levels (b) = 3,364 devices

4 levels (c) = 195,112 devices

5 levels (d) = 11,316,496 devices

9 character topic string supports over 10,000,000 devices in a single deployment

To simplify implementations, wildcards are not supported in MQTT channels.

Topics originating cloud-side start with an uppercase letter

Topics originating deployment-side start with a lowercase letter

In order to minimise the size of messages, paths are hierarchical with single character base58 (to improve readability) elements

e.g. a health message from a device could have the following path/channel "h/1/2/5/B"

## MQTT - and portability

### Uppercase from Platform

### Lower case to(wards) platform

## Coordinator specific messages

## Minimal size & maximum scalability

## Topic hierarchy

## Shared context (platform and device share config)

## No wildcards

# Control messages

## Coordinator sync: Z Topic

- "Z": Between active and "hot-swap" Coordinators, ensures one, and only one, active coordinator per deployment.

This has not been fully considered to date, but would expect to re-use concepts from other active-passive architecture

## On-boarding: O Topic

HIOT devices which do not have an active configuration, e.g. a new device which has just been connected to the deployment’s network, will publish an On-boarding message on topic “o”, containing a unique identifier for that device (typically the MAC address of the active network interface).

All Aggregators subscribe to the On-boarding channel (“o”) and upon receiving an On-boarding message will request a new device path from the Platform by publishing on “o/<agg-path>” where, <agg-path> is the device ID of the first aggregator deployed on the device publishing the message.

The Platform then publishes the new, unique Device ID on topic “O/<agg-path>”. On receipt, of the message, the aggregator forwards it on topic “o/<deviceID>”. Duplicate On-boarding message for the same device, i.e. from other Aggregators, are ignored.



The new device then updates its configuration with the device path provided by the platform.

|  |  |  |
| --- | --- | --- |
| Message sender and Purpose | Path | Data |
| Device: Get a unique device path | “o” | Unique device ID : String |
| Aggregator: Request a new device id | “o/<first agg path>” | Unique device ID: String |
| Platform: set device path | “O/<first agg path>” | Unique device ID: String  Unique device Path: String |
| Aggregator: set device path | “O/<device id>” | Unique device Path: String |

## Device Config: C Topic

Config message:

- "C": Configuration data for the device specified in the rest of the path

- "c": Configuration data from the device to the platform

In order to query the configuration of a specific device, an empty configuration message can be sent from the Platform, the device will then respond with its configuration data.

Below is a typical device configuration message, with comments to aid interpretation

{

"device": {

"hiotId": “”, //String, the unique device ID

"name": “”, //String, user friendly name, assigned via platform

"description": "", //String, description, assigned via platform

"devicePath": "" //String, unique path for the device

},

"roleChannels": { //array of installed roles on the device

"broker": {

"\_id": "", //unique id for this broker

"deployment": "", // unique id for the relevant deployment

"description": "", // a description of what the broker does

"name": "", // name for the broker

"handler": "", // unique ID for the associated handler file

"active": true,

"myPaths": [{ //pairs of relevant subscriptions (“in”) and publications (“out”)

"in": "1/2/3",

"out": "1/2/f1",

"\_id": "59df282a3a092a2b805b656a"

}, {

"in": "1/2/3/4/6/8/#",

"out": "3/3/3/4/",

"\_id": "59df282a3a092a2b805b6569"

}, {

"in": "1/9/f1",

"out": "1/2/3/4/5/6/7/8/9f1",

"\_id": "59df282a3a092a2b805b6568"

}, {

"in": "1/2/3/4/5/6/7/8/9f1",

"out": "1/5/f1",

"\_id": "59df282a3a092a2b805b6568"

}, {

"in": "1/2/f1",

"out": "1/4/f1",

"\_id": "59df282a3a092a2b805b6568"

}, {

"in": "1/4/f1",

"out": "1/1/f1",

"\_id": "59df282a3a092a2b805b6568"

}]

},

"coordinator": { // details of the Coordinator Link to connect to

"m2mMqttport": 0,

"m2mMqttServer": "",

“auth”: {} // relevant authentication parameters

},

"controller": [{

"\_id": "59c393ee54674b1ce4982a69",

"description": "Lighting controller - First Floor",

"name": "LWRF - F1",

"\_\_v": 0,

"deployment": "597f3056ef66be0648ef5bd3",

"controllerId": "cont1203",

"channel": "x/1/1/f1",

"handler": "59bfda6ed7649a3070551ad9",

"commands": {

"1": {

"name": "porch light on/off",

"e": "/home/pi/433Utils/RPi\_utils/codesend 123456"

},

"2": {

"name": "garden light on/off",

"e": "/home/pi/433Utils/RPi\_utils/codesend 654321"

},

"3": {

"name": "kitchen lights on",

"e": "/home/pi/433Utils/lightwave\_rf/send R1S1D1 1 1"

},

"4": {

"name": "kitchen lights off",

"e": "/home/pi/433Utils/lightwave\_rf/send R1S1D1 1 0"

},

"5": {

"name": "sitting room spots on",

"e": "/home/pi/433Utils/lightwave\_rf/send R2S2D1 1 1"

},

"6": {

"name": "sitting room spots off",

"e": "/home/pi/433Utils/lightwave\_rf/send R2S2D1 1 0"

},

"7": {

"name": "sitting room centre on",

"e": "/home/pi/433Utils/lightwave\_rf/send R2S1D1 1 1"

},

"8": {

"name": "sitting room centre off",

"e": "/home/pi/433Utils/lightwave\_rf/send R2S1D1 1 0"

},

"9": {

"name": "send to console",

"e": "msg"

}

}

}],

"aggregator": [{

"\_id": "59db76cdccc548be30e5249f",

"handler": "59bfdae2d7649a3070551ade",

"name": "Aggregator1",

"channel": "a/1",

"description": "some random aggregator",

"poll": 18000,

"deployment": "597f3056ef66be0648ef5bd3",

"active": true,

"\_\_v": 0,

"topics": ["s/1/2/3/4/5/6/7/8/8/9/9/9/9", "s/4/5/22", "s/3/4/5"]

}, {

"\_id": "59db76cdccc548be30e5249a",

"handler": "59bfdae2d7649a3070551adb",

"name": "Aggregator2",

"channel": "a/2",

"description": "some random aggregator",

"poll": 60000,

"deployment": "597f3056ef66be0648ef5bd3",

"active": true,

"\_\_v": 0,

"topics": ["s/1/2/3/4/5/6/7/8/8/9/9/9/9", "s/4/5/22", "a/1/2/3/4/5"]

}],

"sensor": [{

"\_id": "59db98407e3a9fb060064ebf",

"id": "sn1234",

"name": "first real sensor",

"channel": "s/1/2/3/4/5/6/7/8/8/9/9/9/9",

"description": "some random sensor",

"handler": "59db908c327a46cb841ac004",

"poll": 60000,

"config": {

"minValue": 1234,

"range": 100

},

"location": "59bfd3ba9db46f37b858ed29",

"active": true,

"\_\_v": 0

}]

},

"moscaEnabled": true,

"moscaPort": 1883,

"mqttServers": [{

"mqttServerIP": "127.0.0.1",

"mqttServerPort": "1883",

"priority": 1,

"\_id": "59df294d3a092a2b805b6570"

}, {

"mqttServerIP": "10.0.0.1",

"mqttServerPort": "1883",

"priority": 1,

"\_id": "59df294d3a092a2b805b656f"

}]

}

## Health Messages: H Topic

- "H": Platform requests the health status of a particular device in the deployment

- "h": device sends health information to the deployment

-- -- -- move roles when device is short on resources

## Error Messages: E Topic

- "E": Error message from the Platform to a device in a deployment. E.g. Aggregation data is invalid

- "e": Error message from a device to the platform. E.g. a device has not forwarded sensor data to the aggregator

-- -- -- ensure required aggregator and broker roles are active

-- -- -- move roles when device is unavailable

## Handler Messages: N Topic

-- "N": platfrom pushes handler file to a device

# Operational Messages

## Sensor Readings: S Topic

## Execution commands: X Topic

- "X": Platform requests the execution of a specified command on a device (via one or more chained brokers)

- "x": Commander requests the execuyion of a specified command on a device (via a broker)

command:

{

"p":"the path to the device",

"c":"the id of the command to execute,

"p":[

{"p1":"the first parameter"},

...

{"pn":"the nth parameter}

]

}

## Aggregation Results: A topic

- "a","aggregation output from an aggregator, can be sent via the coordinator, or as an input to another aggregator

{

"t":<timestamp>,

[

"1": first data element (e.g. average (mean) value)

"2": second data element (e.g. max)

"D": ["raw" data (from input sensors and/aggregators), if required]

]

}

## Message brokering: B Topic

A Broker message encapsulates the required command message

the path is included so that multiple brokers can be used, e.g.

"p" = "x/5/V/6/4" (device path)

coordinator publishes on "x/5"

Broker subscribed to “x/5”, publishes same message on "x/5/V"

Broker subscribed to “x/5/V”, publishes same message on "x/5/V/6”

Broker subscribed to “x/5/V/6”, publishes **included command message** on "x/5/V/6/4"

device subscribed to "x/5/V/6/4" executes command

## Event Messages: V Topic

- "V": platform notifies deployment/device of an event..??

- "v": device notifies platform of an event (e.g. temperature exceeds predefined limit)

"v" paths include a type as the second element and priority as the third element(e.g. "v/e/1" is a high priority error)

## Query Messages: Q Topic

- "Q": a query from the platform to the device specified in the remainder of the path - to support additional functionality

- "q": a query from a device - to support additional functionality

## Response Messages: R Topic

- "R": Response from the platform - platform has responded to a query from the deployment (requires "smart" roles)

- "r": device responds to the platform - e.g. confirm a role has been moved to a device

# Use cases

## Onboarding a device

## Data aggregation

### Excluding "raw" data

### Including "raw" data

## Command execution from platform

## Command execution from commander

## Updating "health" statistics

## "Balancing" a deployment

## Restoration in the event of a platform failure

Orphaned devices (e.g. where the platform data has become corrupted and the deployment is no longer defined) can recreate the relevant deployment based on their configuration file